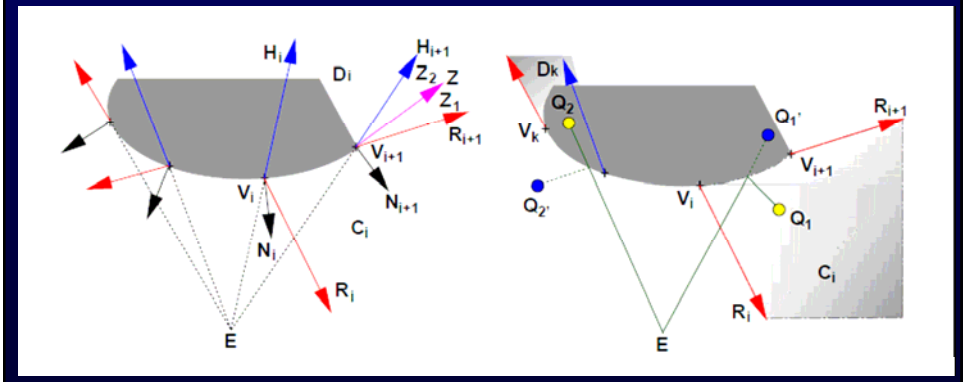
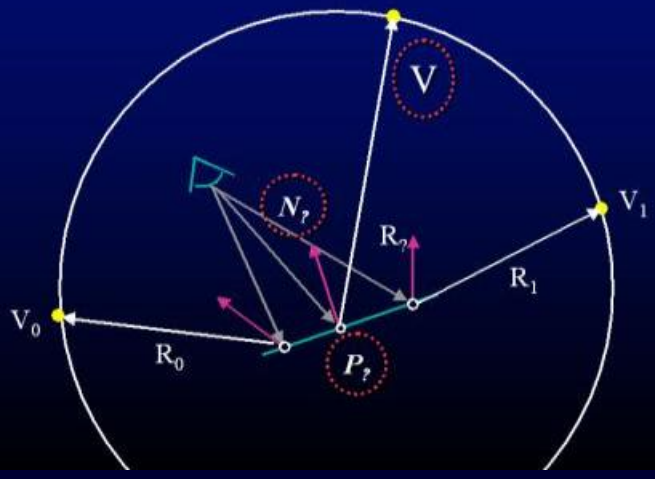
Curved Reflectors



Curved Reflectors

Bound with Sphere

SIGGRAPH



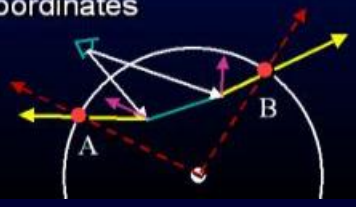
Curved Reflectors

Step 1a: One curved face, restricted reflections

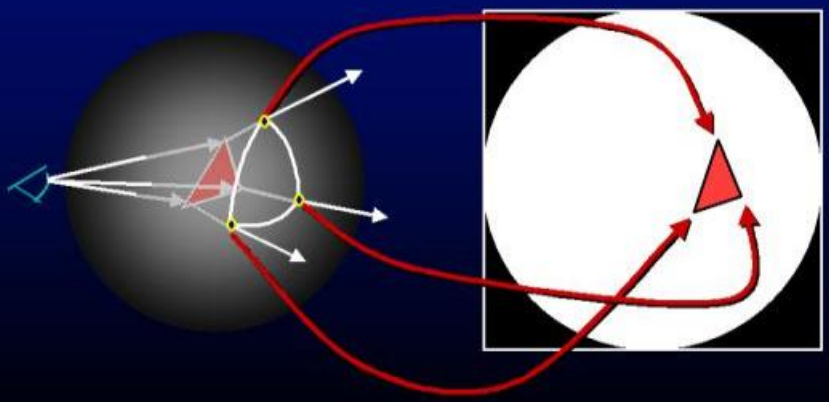
- Find a mapping of surface of sphere to 2D

$$T(x,y,z)=\left[\frac{\frac{1}{2}+x}{\sqrt{x^2+y^2+(z+1)^2}}\quad\frac{\frac{1}{2}+x}{\sqrt{x^2+y^2+(z+1)^2}}\right]$$

- Maps normalized vector from sphere center to point on sphere to 2D coordinates



Curved Reflectors

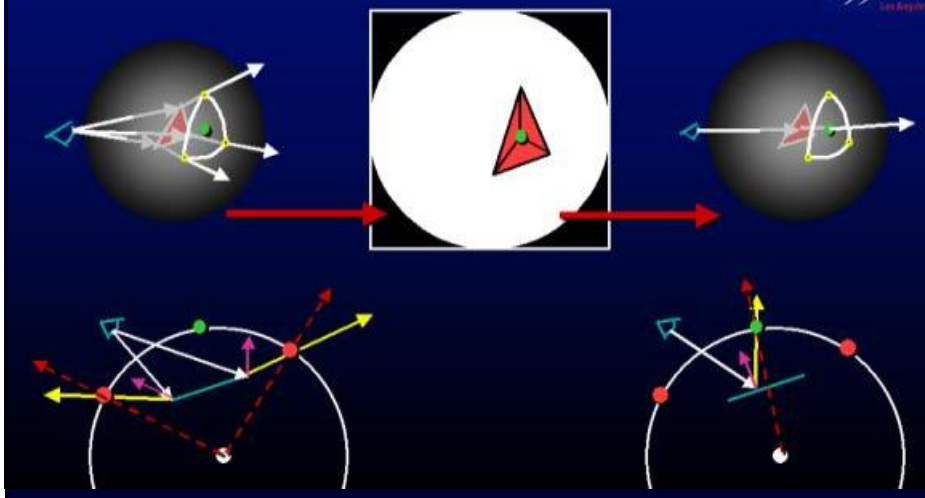


Curved Reflectors

Step 1a: One curved face, restricted reflections

- For any point V on the sphere, find mapping V_{2D}
- Use barycentric weights to interpolate N and P from original vertices of triangle to find N' and P' for V_{2D}
- Use N' and P' to find reflection for V

Curved Reflectors



Algorithm

1. For some vertex Q
2. Find direction (how?)
3. Find the (s,t) for the direction
4. Use the ID to find the polygon
5. Compute barycentric coords in map-space
6. Use barycentric coords to approximate the surface point by interpolating normals
7. Form a plane
   (intersection point, interpolated normal)
8. Reflect the vertex

Issues?

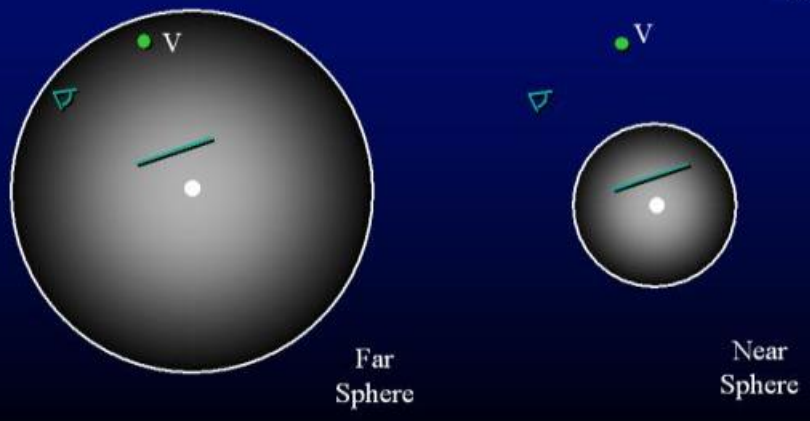
- What are the issues?

Curved Reflectors

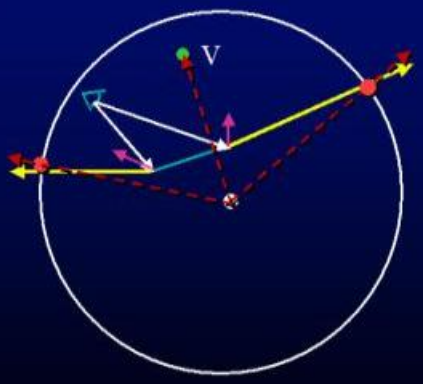
Step 1b: One curved face, arbitrary reflections

- Most scene points won't lie on one sphere
- But could use *two* spheres, approximate, and combine results!
- "Near" sphere tightly bounds reflector (between reflector and other objects)
- "Far" sphere loosely bounds the whole scene(contains scene and reflector)
- Perform reflections as before, but then blend reflected V'

Curved Reflectors



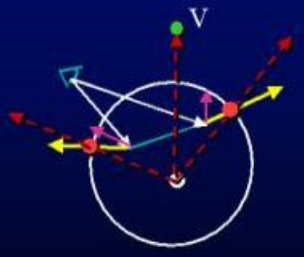
Curved Reflectors



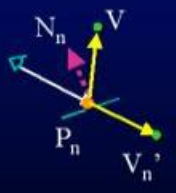
Interpolated

"Far" Sphere Loosely Bounds the Entire Scene

Curved Reflectors



Interpolated



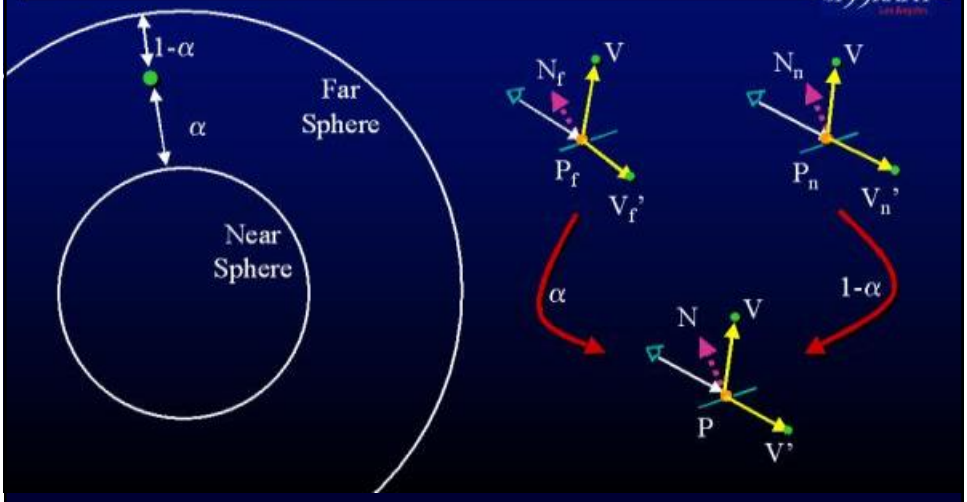
"Near" Sphere Loosely Bounds the Entire Scene

Curved Reflectors

Step 1b: One curved face, arbitrary reflections

- Find reflection of V using barycentric weights from near sphere and 2D mapping
- Then find reflection of V using far sphere
- Use distance of V between spheres to blend the two reflections

Curved Reflectors

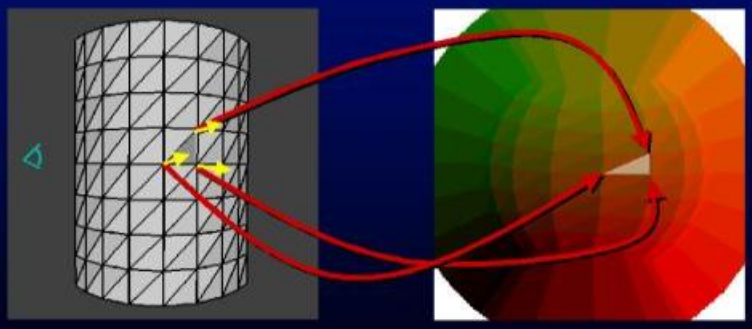


Curved Reflectors

Step 2: Mesh of curved faces

- With multiple faces, how to choose which one gives interpolation weights, etc?
- Use "explosion map"
  - Each triangle in mesh mapped to 2D
  - Triangle's "ID" colored into 2D image
  - Forms a lookup table
  - Not a texture map; *used for CPU lookup only*

Curved Reflectors



Curved Reflectors

Using an explosion map

- Ofek & Rappoport say 200x200 seems to suffice
- Will need to extend mapped polygons to the edge
- Partly backfacing "profile" faces need to have edges extended to at least .6 of radius of map

Curved Reflectors

Tessellation issues

- Need to tessellate real objects to reduce curvature error
- Need to tessellate reflector to reduce interpolation error
- Good news is that the eye is just looking for edges and appearance of reflected *motion* and *contours*

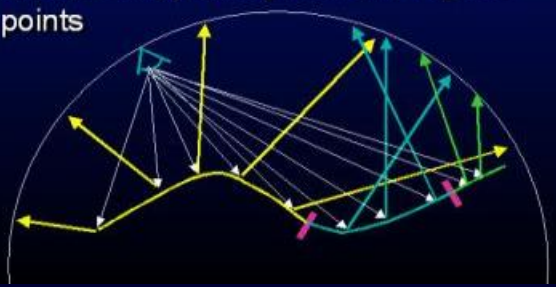
Demo

Curved Reflections

SIGGRAPH

Concave reflectors

- This technique assumed F(R) was unique; only convex objects have that property
- But can dice up concave and partially concave objects around inflection points

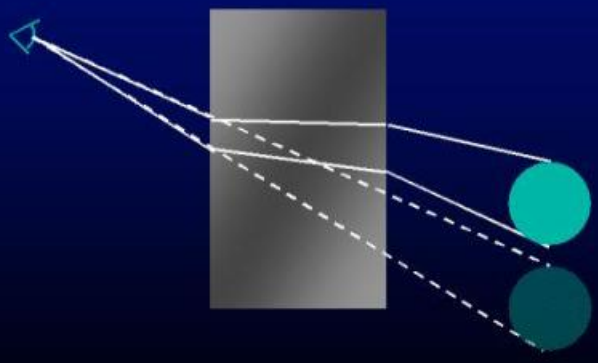


Refraction

Transmission of Light

- Extend notion of "virtual" scene from reflection to *refraction*
- Think of refracted scene as original scene with some transformation applied
- Calculate transform, apply, clip results to front *and back* faces of refractor

Refraction



But lots of problems with this: circle of confusion, etc

A GPU-driven Algorithm for Accurate Interactive Reflections on Curved Objects

Pau Estalella, Ignacio Martin,
George Drettakis, Dani Tost

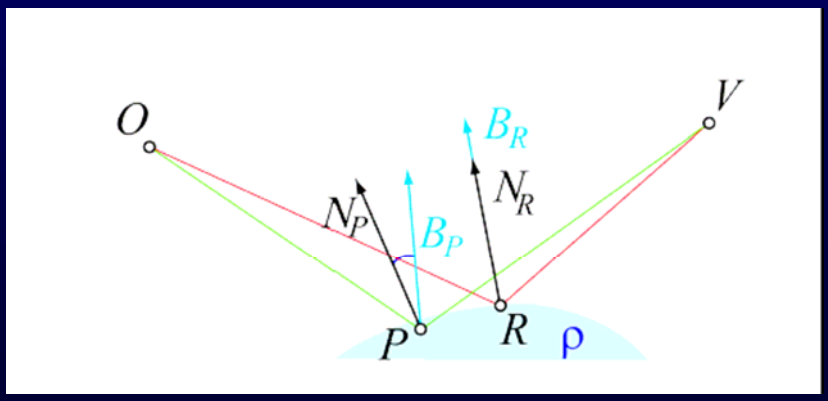
Basic Algorithm

foreach frame

- computeReflectedScenes()
- drawNonReflectors()
- drawReflectorsWithStencil()
- drawReflectedScenes()

endfor

Basic Idea



Use Search in Reflector Image

- Cache reflector vertex and reflector normal

computeReflectedScenes{

foreach reflector Ri

1. renderAndStore3DandNormalTextures()
2. setUpRenderTargets()
3. setUpCg()
4. sendVerticesToGPU()
5. copyResultToReflectedVertexArray()

Endfor

}

Foreach virtual vertex

- Search the reflector maps

- Look for N dot B = 1 (or maximum)

Results:

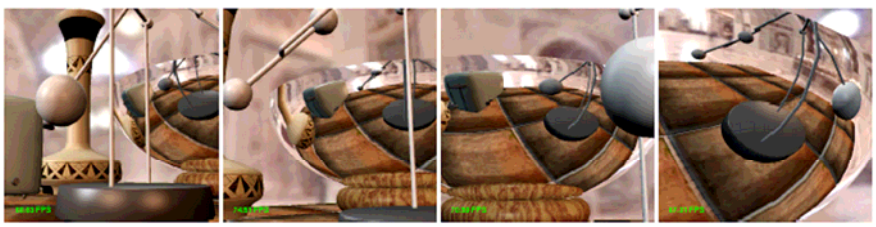


Figure 7: *Four images of the kitchen scene. This scene runs at 80 fps and contains 14,979 vertices.*

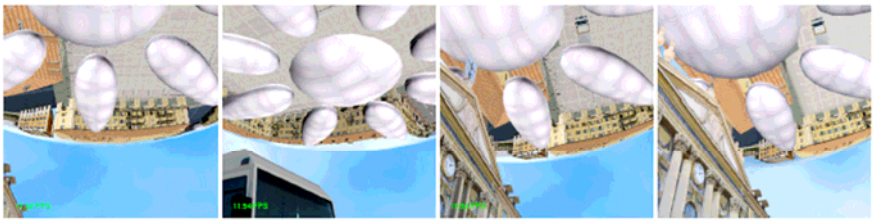
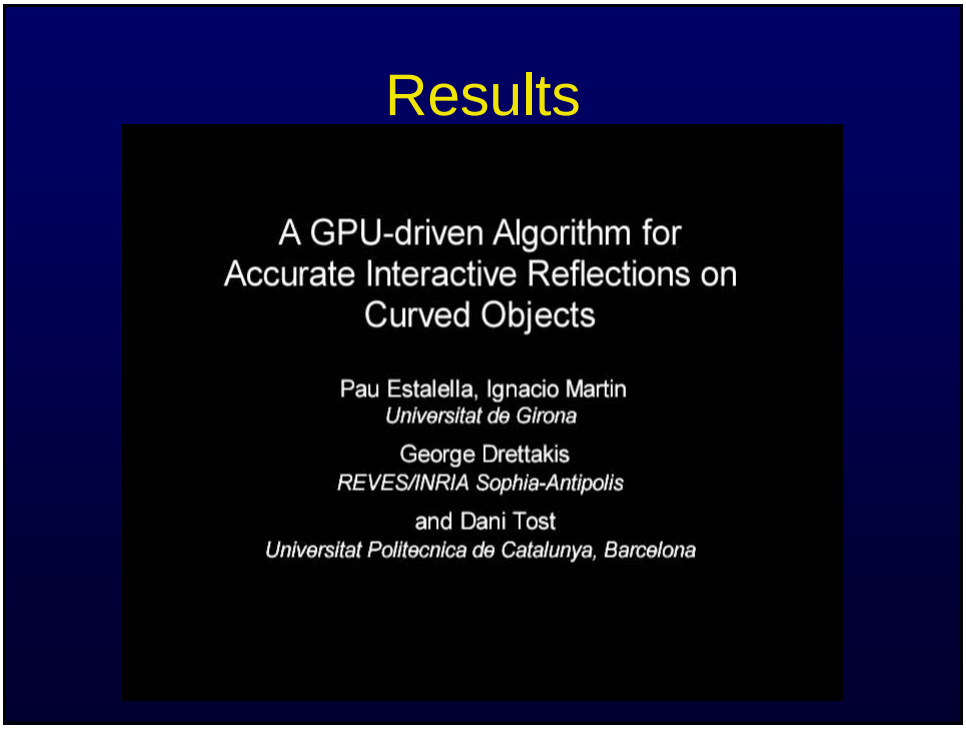


Figure 8: *(Left) Three frames of the UFO scene (12 fps; 81,997 vertices). (Far right) Ray-traced image for comparison.*

Results


A GPU-driven Algorithm for
Accurate Interactive Reflections on
Curved Objects
Pau Estalella, Ignacio Martin
Universitat de Girona
George Drettakis
REVES/INRIA Sophia-Antipolis
and Dani Tost
Universitat Politecnica de Catalunya, Barcelona